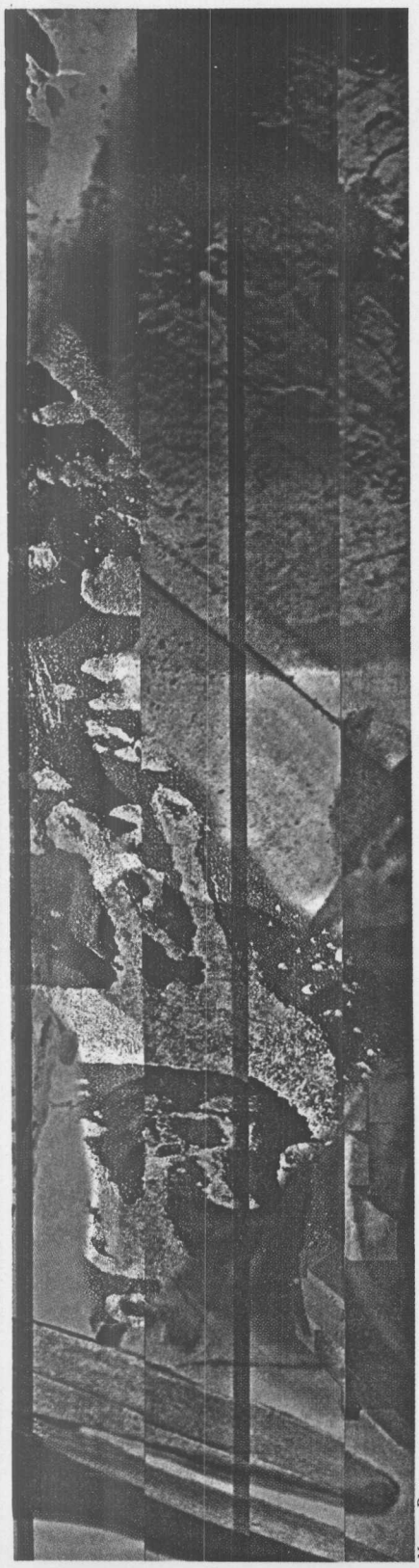
by DINU MADAU



Nance Paternoster

Influence Value Defuzzification Method

Defuzzification is compute intensive and may seriously bog down your system. Here is a computationally efficient method that reduces the resources required and eliminates the need for a faster microcontroller or custom logic.

You have been given the task to design a fuzzy logic embedded system. At the heart of the system is a very "cost-effective" microcontroller. You and I know that cost-effective microcontroller system translates into "s-l-o-w." Now suppose that you are to design a system that requires update rates on the order of milliseconds (control systems with fast actuators could fall into this category). You spend a month or so putting the project together, everything is going great, and then you realize that the defuzzification routine in your fuzzy logic module is eating up all of your loop time. In essence, the thing is too slow. Of course, you knew this at the beginning of the project. You first realize that a means of reducing the burden on the microcontroller is, naturally, special computational hardware, such as custom integrated circuits; such a solution, while attractive, increases both complexity and cost. Your second option is to switch to a faster microcontroller. So you tell your management that you need a faster (and more expensive) microcontroller and they quickly remind you that adding a dime to the cost of the microcontroller would result in an increased project cost of $100,000 per year (which, by the way, is significantly more than you get paid). You quickly reconsider the second option. The third option is to dispense with max-min inference and instead employ a more computationally favorable method of inference. An increasingly popular approach is the use of symmetric output functions (such as gaussians) in combination with output aggregation

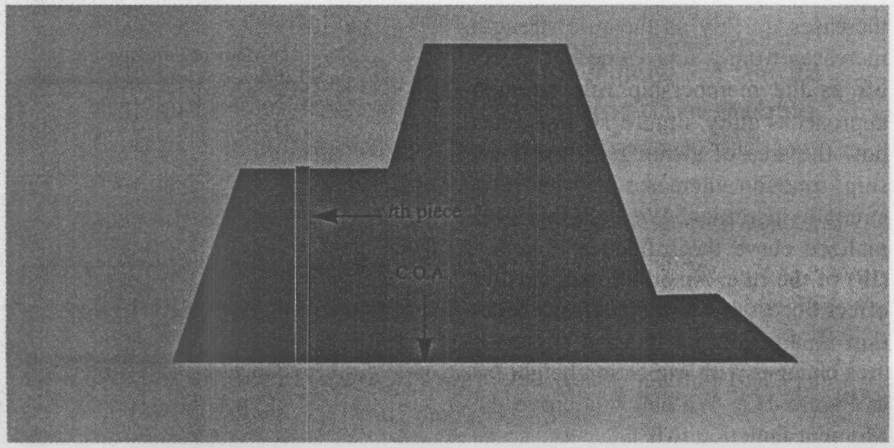
We realized an estimated 94% reduction of defuzzification computation time, compared to the piece-wise integration technique of COA.

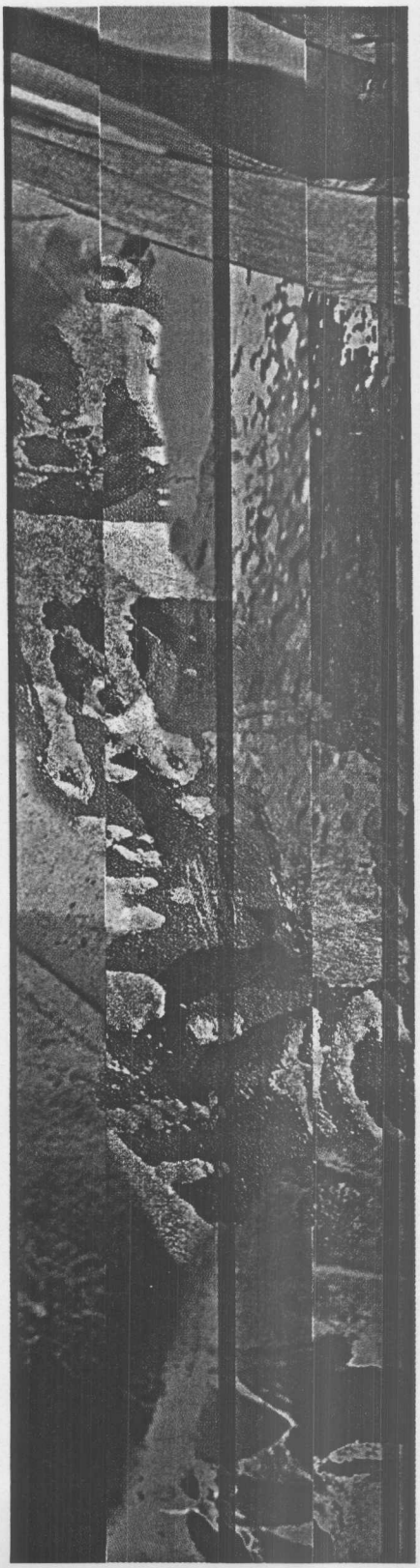
with PLUS rather than MAX. This approach allows the output functions to be characterized on the basis of their areas and centroids and avoids entirely the need for numerical integration in the defuzzification process.

As you consider these options, you determine that for your particular application, you need the true center of area (COA) defuzzification. I don't want to debate your design choice or the relative merits of the various inference methods. Rather, I will assume that max-min inference is deemed preferable for certain applications. Fortunately, a computationally efficient method of defuzzification, which we call the influence value (IV) algorithm, exists. The IV algorithm reduces the resources necessary for computation of the COA defuzzification. To whet your appetite, I submit a project that I was involved in: we realized an estimated 94% reduction of defuzzification computation time, when compared to the full piece-wise integration technique of COA. For practical applications, this reduction permits max-min inference to be carried out with inexpensive processing devices at relatively high speeds.

I am making the assumption that you have some basic understanding of fuzzy logic, inference methods, and the COA defuzzification technique. If not, don't worry: this method relates only to the COA output defuzzification, which mathematically translates to determining the center of area of a given centroid. So before we go any further, here's a brief review on how to get the center of area of a centroid.

FIGURE A
Output centroid.





CENTER OF AREA

Given a set of fuzzy logic rules and their associated output memberships, the combination of the truncated output memberships (typically triangular or trapezoidal) generates an output centroid whose center of area is the crisp output result of the fuzzy logic computation (see Figure A).

The center of area requires the zeroth and first moments of the output function. The first moment of the centroid is divided by the zeroth moment to generate a crisp result. In fixed-point processors, a straightforward defuzzification procedure would typically be based on a discretized output range. The range, for example, could be a gain for your controller, an on-time for a solenoid, or whatever you desire to control or manipulate. The area is calculated by piece-wise integration of the centroid. For each piece in the range, the value derived from the `MAX` operation is the height for the piece that is then used to determine its area. The area multiplied by the relative position of the piece in the range yields its moment with respect to the origin. The crisp output is computed with one multiply and two additions for each element in the range, followed by a final division operation. The output equation is then given by:

$$COA = \frac{\Sigma_i[\text{moment}(i)]}{\Sigma_i[\text{area}(i)]}$$

where *i* indexes the discretized pieces across the output range. Therefore, the computation time scales roughly with the number of values into which the output range is divided.

MAX-MIN INFERENCE

Assuming that you have an understanding of max-min inference, I will discuss only those aspects necessary to explain the present method. In particular, the details by which rule strengths are obtained from inputs and input membership functions are not considered here. I need only assert that a strength has been computed for each rule and is applied to the corresponding output membership function. Secondly, the output membership function is truncated at the rule strength, although the method may be applied if the output function is scaled rather than truncated. In fixed-point processors, the height of the output membership functions is an integer value (derived from the rule strength for that membership with a range typically 0 to 1.0). The value is scaled dependent on the resolution required by the algorithm designer. For example, {0...1.0} could translate to {0...64} where 64 is equivalent to 1.0. Having stated the previous, let's concentrate on the mechanics of defuzzification that require determination of a crisp output value, using agglomeration by `MAX` and computing the center of area, from the set of output membership functions, typically of triangular or trapezoidal shape, and from the set of rule strengths.

We may make some simple observations about the mechanics of defuzzification on max-min inference. First, the strength of a rule influences both the numerator and denominator in the COA calculation, but in neither case is such influence a linear function of the strength of the rule. The primary reason for this, of course, is that the area of a truncated triangle or trapezoid is a nonlinear function of the height of truncation. For example, a triangular membership function's influence increases quickly as the rule strength increases from a low value but tapers off as the membership rule strength approaches unity. Figure 1B illustrates how the area of a triangular membership function increases as the rule-strength increases. We call this normalized curve the Influence Function (IF) of the rule. An additional, smaller effect occurs if the membership function is asymmetric: its own center of area changes with truncation height (as in Figure 1C). We call this curve the Moment Function (MF).

FIGURE 1A
Output membership function.

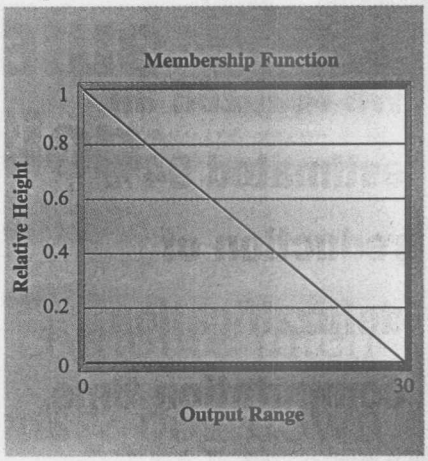


FIGURE 1B
Relationship of the change in membership area vs. rule strengths.

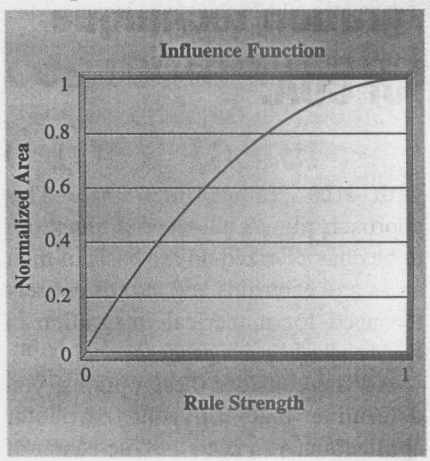
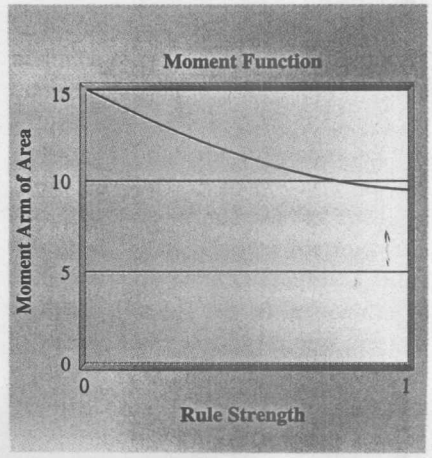


FIGURE 1C
Relationship of the change in membership moments vs. rule strengths.



Another observation is that the contribution of a particular rule on the numerator and denominator in the COA calculation depends on the strength of neighboring rules, such as those rules whose output membership functions overlap the rule in question. Simply stated, the use of MAX agglomeration means that if membership functions overlap, the overlapping areas contribute only once. (This interaction between the rules is a principal feature of the max-min inference—though it frustrates simple superposition, the interaction has the effect of discounting redundant evidence and is appropriate in certain circumstances.)

INFLUENCE VALUE (IV) DEFUZZIFICATION METHOD

The intent in devising the IV method was to formulate a defuzzification scheme whose computation scales more closely with the number of rules than with the number of output divisions. The non-linearities mentioned above can be handled in an obvious way—that is, by precalculating the variations in areas and centroids for each rule as a function of rule strength and storing them in lookup tables. In fact, we may precalculate the moments themselves and store these instead of the centroids. When all is said and done, we will have generated two tables, an area table and a moment table, which will be indexed by the rule strength for the unique output memberships. For fixed-point implementation, one would typically choose a convenient number of values of rule strength, such as a power of 2. Using such tabulations, we could perform an approximate defuzzification by performing two summations over rules, indexed by *i*. The numerator of the COA is approximated as:

$$\Sigma[\text{moment}(i)]$$

and the denominator as:

$$\Sigma[\text{area}(i)]$$

where the values *moment(i)* and *area(i)* depend on the strength of rule *i*.

Though this approximation does not take proper account of the possible overlap between output membership functions, an elegant way of doing this exists. The idea is to augment the rule base with pseudo-rules which will compensate for the overlap regions. Let us suppose, at first, that output membership functions overlap only in pairs. We create a pseudo-rule for each overlapping pair of output membership functions. The output membership function of a pseudo-rule is formed from the overlap region of the original output membership functions. The

moments for the new pseudo-rule will be positive if those of the overlapped pseudo-rule were negative. Figure 3 is an example in which three membership functions overlap. In this case four pseudo-rules would be created.

Table 1 summarizes the 7 output functions, the signs to be associated with their areas and moments, and the rule strengths to be used in the table lookup process.

Note that the sign for pseudo-rule 4 (p4) is positive because it is generated by the overlap of two pseudo-rules (rules p2 and p3, whose signs are negative). It should be emphasized that the work of constructing the pseudo-rules and calculating the tables for moments and areas needs to be performed only once for a given rule base. For our project, we wrote a program that automatically generates the two tables, given a set of triangular/trapazoidal output memberships. Because the computational cost of defuzzification roughly scales with the total number of rules and pseudo-rules, the present method is best applied to systems in which the overlap patterns are not too complex. Furthermore, it may be satisfactory to ignore pseudo-rules whose membership functions have very little area, in direct analogy with truncating a series expansion when the terms become small.

CRISP OUTPUT

Table 2 illustrates the IVD tables for the example in Figure 3 with the following points for vertices:

a = 0
b = 0
c = 30
d = 15
e = 30
f = 60
g = 25
h = 64
i = 64

The least significant bit of rule-strength is equal to 0.1 (0 to 10 equivalent 0 to 1.0). The following pseudo-code can be used to generate a crisp output, given rule-strengths for the individual rules (`A_Table` = area table and `M_Table` = moment table):

```
Area = 0;
Moment = 0;
RuleNumber = 1;

while (RuleNumber <= MaxNoOfRules)
  {
                Area                    +=
A_Table(RuleNumber,RuleStrength);
              Moment                    +=
M_Table(RuleNumber,RuleStrength);
  }

COA = Moment / Area;
```

TABLE 2
Influence value lookup tables for the example using 10 intervals for rule-strengths (note p4 areas and moments would be positive).

Area Table

RS:	0	1	2	3	4	5	6	7	8	9	10
Rule											
1	0	28	54	76	96	112	126	136	144	148	150
2	0	42	81	114	144	168	189	204	216	222	225
p1	0	-12	-21	-24	-25	-25	-25	-25	-25	-25	-25
3	0	37	70	99	124	146	163	177	187	193	195
p3	0	-31	-56	-73	-84	-88	-88	-88	-88	-88	-88
p2&p4	0	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

Moment Table

RS:	0	1	2	3	4	5	6	7	8	9	10
Rule											
1	0	406	732	985	1176	1312	1404	1459	1488	1498	1500
2	0	1587	2979	4181	5202	6046	6723	7237	7596	7806	7875
p1	0	-282	-459	-536	-541	-541	-541	-541	-541	-541	-541
3	0	1684	3255	4699	5999	7141	8110	8890	9466	9822	9945
p3	0	-1347	-241	-3185	-3663	-3839	-3840	-3840	-3840	-3840	-3840
p2&p4	0	-50	-50	-50	-50	-50	-50	-50	-50	-50	-50

Finally, it all comes down to time and space. So let's go where not everyone has gone before and determine the requirements on system resources.

TIME

Given equivalent performance requirements, the computation time for IVD is less than that of standard COA, which requires you to derive the area of the centroid by piece-wise integration and multiply each piece by its position to get the moment. To quantify the difference, I used a data book for a Texas Instruments microcontroller to determine the number of clock cycles required to execute both algorithms (COA with 64 intervals in the output and IVD with 10 divisions of rule strength). The following instructions were used to estimate the cycles necessary to execute the core COA and IV defuzzification algorithm:

- Max operation (Table Lookup): 4 cycles
- Multiplication: 16 cycles
- Addition: 1 cycle
- Division: 29 cycles

Let's focus our attention to the COA method. For 64 intervals in the output range, you need 64 lookups to determine the Max of each interval, 64 multiplications to calculate the moment of each interval and 128 additions to sum up the moments and areas, plus one final division to get the crisp output. This requires 1309 clock cycles. The IVD method requires 14 table lookups to get the moment and area for each rule/pseudo-rule, 14 additions for the summation of each rule's moment and area, and one division for the crisp output for a total 99 clock cycles—a reduction in clock cycles of 94%.

SPACE

The sizes of the data tables depend on the number of intervals for the rule strength and the number of rules/pseudo rules. For 10 intervals of rule strength, Table 2 is generated for the example in Figure 3. Again, please note that the area and moment arm for p2 and p4 are relatively small when compared to the other memberships. Removing p2 and p4 from the tables may be appropriate due to these rules' relatively small influence on the output. Also note that the size of these tables are quite modest when compared to direct input/output map tables.

IVD

Influence Value Defuzzification algorithms significantly reduce the computational cost associated with the final steps in standard max-min (Mamdani) inference: aggregating the output membership functions and computing the center of area. In contrast to a straightforward implementation of max-min inference, the present method does not rely on a discretization of the output range, though it does require a discretization of the range of rule strength. The accuracy of the computed center of area, as well as the storage space required for the required lookup tables, increases as this range is discretized more finely.

Although I have concentrated on output membership functions of triangular or trapezoidal shape, the method can be applied to more general shapes without unduly increased on-line computational cost of defuzzification. If the overlap regions become more complicated, the complexity of computing (off-line) the required lookup tables (such as Table 2) will naturally increase. The practical significance of this method is that it permits max-min inference to be executed on inexpensive processors at relatively high speed. ESP

Dinu P. Madau is a product design engineer with Ford Motor Company. He has worked with embedded systems for 10 years. He is responsible for software development in the Chassis Electronic Systems department of Advanced Vehicle Technology. He received his undergraduate degree from Oakland University and his graduate degree in Electronic and Computer Control Systems from Wayne State University. Madau can be reached electronically at dmadau@ford.com.